Segmentation of Indian Traffic

```
#Run only if necessary (Incase of DLL error)
# Check libcudnn8 version
!apt-cache policy libcudnn8

# Install latest version
!apt install --allow-change-held-packages libcudnn8=8.4.1.50-1+cuda11.6

# Export env variables
!export PATH=/usr/local/cuda-11.4/bin${PATH:+:${PATH}}
!export LD_LIBRARY_PATH=/usr/local/cuda-11.4/lib64:$LD_LIBRARY_PATH
!export LD_LIBRARY_PATH=/usr/local/cuda-11.4/include:$LD_LIBRARY_PATH
!export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/cuda/extras/CUPTI/lib64

# Install tensorflow
!pip install tflite-model-maker==0.4.0
!pip uninstall -y tensorflow && pip install -q tensorflow==2.9.1
!pip install pycocotools==2.0.4
!pip install opencv-python-headless==4.6.0.66
```

```
Building dependency tree
Reading state information... Done
libcudnn8 is already the newest version (8.4.1.50-1+cuda11.6).
0 upgraded, 0 newly installed, 0 to remove and 22 not upgraded.
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting tflite-model-maker==0.4.0
  Using cached tflite_model_maker-0.4.0-py3-none-any.whl (642 kB)
Collecting matplotlib<3.5.0,>=3.0.3
  Using cached matplotlib-3.4.3.tar.gz (37.9 MB)
  Preparing metadata (setup.py) ... done
Requirement already satisfied: tensorflow-datasets>=2.1.0 in /usr/local/lib/python3.10/dist-packages (from tflite-model-maker==0.
Requirement already satisfied: Cython>=0.29.13 in /usr/local/lib/python3.10/dist-packages (from tflite-model-maker==0.4.0) (0.29.
Requirement already satisfied: pillow>=7.0.0 in /usr/local/lib/python3.10/dist-packages (from tflite-model-maker==0.4.0) (8.4.0)
Collecting tensorflow-addons>=0.11.2
  Using cached tensorflow_addons-0.20.0-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (591 kB)
Collecting tensorflowjs>=2.4.0
  Downloading tensorflowjs-4.4.0-py3-none-any.whl (85 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 85.1/85.1 kB 6.7 MB/s eta 0:00:00
Requirement already satisfied: absl-py>=0.10.0 in /usr/local/lib/python3.10/dist-packages (from tflite-model-maker==0.4.0) (1.4.0
Collecting tensorflow-hub<0.13,>=0.7.0
  Using cached tensorflow_hub-0.12.0-py2.py3-none-any.whl (108 kB)
Requirement already satisfied: PyYAML>=5.1 in /usr/local/lib/python3.10/dist-packages (from tflite-model-maker==0.4.0) (6.0)
Collecting fire>=0.3.1
  Downloading fire-0.5.0.tar.gz (88 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 88.3/88.3 kB 8.3 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
Collecting tensorflow-model-optimization>=0.5
  Downloading tensorflow_model_optimization-0.7.4-py2.py3-none-any.whl (240 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 240.6/240.6 kB 18.5 MB/s eta 0:00:00
Collecting neural-structured-learning>=1.3.1
  Using cached neural_structured_learning-1.4.0-py2.py3-none-any.whl (128 kB)
Collecting librosa==0.8.1
  Downloading librosa-0.8.1-py3-none-any.whl (203 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 203.8/203.8 kB 23.0 MB/s eta 0:00:00
Requirement already satisfied: lxml>=4.6.1 in /usr/local/lib/python3.10/dist-packages (from tflite-model-maker==0.4.0) (4.9.2)
ERROR: Ignored the following versions that require a different python version: 0.52.0 Requires-Python >=3.6,<3.9; 0.52.0rc3 Requi
ERROR: Could not find a version that satisfies the requirement numba==0.53 (from tflite-model-maker) (from versions: 0.1, 0.2, 0.
ERROR: No matching distribution found for numba==0.53
Found existing installation: tensorflow 2.9.1
Uninstalling tensorflow-2.9.1:
  Successfully uninstalled tensorflow-2.9.1
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: pycocotools==2.0.4 in /usr/local/lib/python3.10/dist-packages (2.0.4)
Requirement already satisfied: matplotlib>=2.1.0 in /usr/local/lib/python3.10/dist-packages (from pycocotools==2.0.4) (3.7.1)
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (from pycocotools==2.0.4) (1.22.4)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=2.1.0->pycocotools==2
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=2.1.0->pycocotools==
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=2.1.0->pycocotools==2.0
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=2.1.0->pycocotools=
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=2.1.0->pycocotoc
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=2.1.0->pycocotools==
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=2.1.0->pycocotools=
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=2.1.0->pycocotools==2.0.
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.7->matplotlib>=2.1.0-
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: opencv-python-headless==4.6.0.66 in /usr/local/lib/python3.10/dist-packages (4.6.0.66)
Requirement already satisfied: numpy>=1.14.5 in /usr/local/lib/python3.10/dist-packages (from opencv-python-headless==4.6.0.66) (
```

```
import math
from PIL import Image, ImageDraw
from PIL import ImagePath
```

```
from PIL import ImageFath
import pandas as pd
import os
from os import path
from tqdm import tqdm
import json
import cv2
import numpy as np
import matplotlib.pyplot as plt
import urllib
```

```
!apt install --allow-change-held-packages libcudnn8=8.1.0.77-1+cuda11.2
```

```
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages will be REMOVED:
  libcudnn8-dev
The following held packages will be changed:
  libcudnn8
The following packages will be DOWNGRADED:
  libcudnn8
0 upgraded, 0 newly installed, 1 downgraded, 1 to remove and 22 not upgraded.
Need to get 430 MB of archives.
After this operation, 1,153 MB disk space will be freed.
Get:1 https://developer.download.nvidia.com/compute/cuda/repos/ubuntu2004/x86_64  libcudnn8 8.1.0.77-1+cuda11.2 [430 MB]
Fetched 430 MB in 6s (74.6 MB/s)
(Reading database ... 122349 files and directories currently installed.)
Removing libcudnn8-dev (8.7.0.84-1+cuda11.8) ...
update-alternatives: removing manually selected alternative - switching libcudnn to auto mode
dpkg: warning: downgrading libcudnn8 from 8.7.0.84-1+cuda11.8 to 8.1.0.77-1+cuda11.2
(Reading database ... 122316 files and directories currently installed.)
Preparing to unpack .../libcudnn8_8.1.0.77-1+cuda11.2_amd64.deb ...
Unpacking libcudnn8 (8.1.0.77-1+cuda11.2) over (8.7.0.84-1+cuda11.8) ...
Setting up libcudnn8 (8.1.0.77-1+cuda11.2) ...
```

Preprocessing

1. Get all the file name and corresponding json files

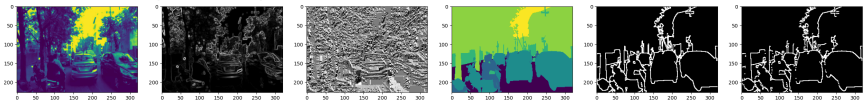
```
import tarfile

file = tarfile.open('/path/to/final_thesis_dataset/idd-lite.tar.gz')

file.extractall('/content/dataset')

file.close()
```

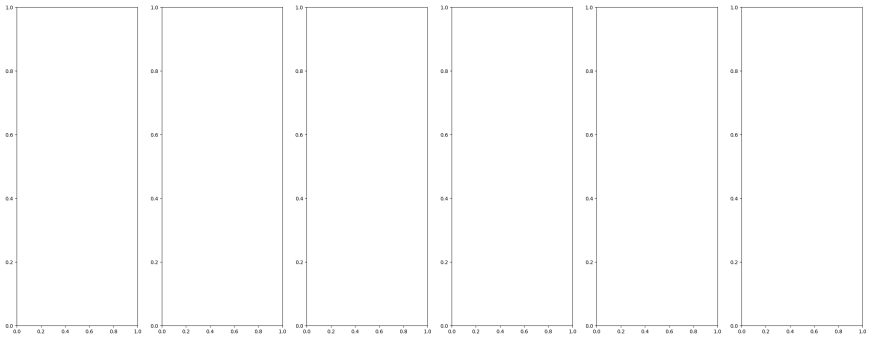
```
def return_file_names_df(root_dir):
    """Function to return an array consisting of path of each of the original image and labels for both train and evaluation da
    data_root_path=root_dir
    images_doc='leftImg8bit'
    mask_doc='gtFine'
    image_path=os.path.join(data_root_path,images_doc)
    train_image_path_scenes=os.listdir(os.path.join(image_path,'train'))
    val_image_path_scenes=os.listdir(os.path.join(image_path,'val'))
    mask_path=os.path.join(data_root_path,mask_doc)
    train_mask_path_scenes=os.listdir(os.path.join(mask_path,'train'))
    val_mask_path_scenes=os.listdir(os.path.join(mask_path,'val'))
    train_images_data_path=[]
    train_mask_data_path=[]
    for i in range(len(train_image_path_scenes)):
      scene=train_image_path_scenes[i]
      image_path_scene=os.path.join(image_path+'/train',scene)
      mask_path_scene=os.path.join(mask_path+'/train',scene)
      image_frames=os.listdir(image_path_scene)
      mask_frames=os.listdir(mask_path_scene)
      frames=[frame.split('_')[0] for frame in image_frames]
      image_frames='_image.jpg'
      mask_frames='_label.png'
      for indiv_frame in frames:
        image_frame_concat=indiv_frame+image_frames
        mask_frame_concat=indiv_frame+mask_frames
        train_images_data_path.append(os.path.join(image_path_scene,image_frame_concat))
        train_mask_data_path.append(os.path.join(mask_path_scene,mask_frame_concat))

    val_images_data_path=[]
    val_mask_data_path=[]
    for i in range(len(val_image_path_scenes)):
      scene=val_image_path_scenes[i]
      image_path_scene=os.path.join(image_path+'/val',scene)
      mask_path_scene=os.path.join(mask_path+'/val',scene)
      image_frames=os.listdir(image_path_scene)
      mask_frames=os.listdir(mask_path_scene)
      frames=[frame.split('_')[0] for frame in image_frames]
      image_frames='_image.jpg'
      mask_frames='_label.png'
      for indiv_frame in frames:
        image_frame_concat=indiv_frame+image_frames
        mask_frame_concat=indiv_frame+mask_frames
        val_images_data_path.append(os.path.join(image_path_scene,image_frame_concat))
        val_mask_data_path.append(os.path.join(mask_path_scene,mask_frame_concat))

    train_data=pd.DataFrame(list(zip(train_images_data_path,train_mask_data_path)),columns=['image','mask'])
    test_data=pd.DataFrame(list(zip(val_images_data_path,val_mask_data_path)),columns=['image','mask'])
    return train_data,test_data


root_dir='/content/dataset/idd20k_lite'
train_data,test_data = return_file_names_df(root_dir)
print(train_data.head())
print(test_data.head())
```

```
                                               image  \
0  /content/dataset/idd20k_lite/leftImg8bit/train...
1  /content/dataset/idd20k_lite/leftImg8bit/train...
2  /content/dataset/idd20k_lite/leftImg8bit/train...
3  /content/dataset/idd20k_lite/leftImg8bit/train...
4  /content/dataset/idd20k_lite/leftImg8bit/train...

                                                mask
0  /content/dataset/idd20k_lite/gtFine/train/411/...
1  /content/dataset/idd20k_lite/gtFine/train/303/...
2  /content/dataset/idd20k_lite/gtFine/train/117/...
3  /content/dataset/idd20k_lite/gtFine/train/117/...
4  /content/dataset/idd20k_lite/gtFine/train/213/...
                                               image  \
0  /content/dataset/idd20k_lite/leftImg8bit/val/6...
1  /content/dataset/idd20k_lite/leftImg8bit/val/4...
2  /content/dataset/idd20k_lite/leftImg8bit/val/4...
3  /content/dataset/idd20k_lite/leftImg8bit/val/5...
4  /content/dataset/idd20k_lite/leftImg8bit/val/5...

                                                mask
0  /content/dataset/idd20k_lite/gtFine/val/66/831...
1  /content/dataset/idd20k_lite/gtFine/val/418/00...
2  /content/dataset/idd20k_lite/gtFine/val/418/00...
3  /content/dataset/idd20k_lite/gtFine/val/555/fr...
4  /content/dataset/idd20k_lite/gtFine/val/555/fr...
```

```
from PIL import Image
import numpy as np
import imgaug.augmenters as iaa

#Sample of image, image'smagnitude, image's oreintation, label, label's magnitude, label's oreintation

fig,ax=plt.subplots(2,6,figsize=(30,25))

aug_ed=iaa.EdgeDetect(alpha=1)
img_path=train_data['image'][3]
mask_path=train_data['mask'][3]
img=cv2.imread(img_path)
img_mask=cv2.imread(mask_path)
img_mask[img_mask==255]=0
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)/255.0
gray_mask=img_mask[:,:,1]

#Gradient for image
gX = cv2.Sobel(gray, cv2.CV_64F, 1, 0)
gY = cv2.Sobel(gray, cv2.CV_64F, 0, 1)

#gradient for mask
gX_mask = cv2.Sobel(gray_mask, cv2.CV_64F, 1, 0)
gY_mask = cv2.Sobel(gray_mask, cv2.CV_64F, 0, 1)

#Equal Weightage for both the gradients
combined = cv2.addWeighted(gX, 0.5, gY, 0.5, 0)

# compute the gradient magnitude and orientation
#for image
magnitude = np.sqrt((gX ** 2) + (gY ** 2))
orientation = np.arctan2(gY, gX) * (180 / np.pi) /180
#for labels
magnitude_mask = np.sqrt((gX_mask ** 2) + (gY_mask ** 2))
orientation_mask = np.ceil(abs(np.arctan2(gY_mask, gX_mask) * (180 / np.pi) /180))
#If the magnitude of the labels is greater than 0 then the value is set to 1
#This is because magnitude shows the strength of the change in the image. Incase of labels there will be change only when the pi
#Therby the value increases only on the edges between the class and the rest part will be 0. Therfore setting 1 to all values th
magnitude_mask[magnitude_mask>0] = 1



ax[0][0].imshow(gray)
ax[0][1].imshow(magnitude, cmap="gray")
ax[0][2].imshow(orientation, cmap="gray")
ax[0][3].imshow(gray_mask)
ax[0][4].imshow(magnitude_mask, cmap='gray')
ax[0][5].imshow(orientation_mask, cmap='gray')
```

```
<matplotlib.image.AxesImage at 0x7fe96f4c8f40>
```





2. Dataset Preperation and Augmentation

```
#Creating Directory to save the prepared dataset to be able to load conviniently from the location when loading from DataLoader
import os
os.mkdir("/content/processed_data")
os.mkdir("/content/processed_data/train_data")
os.mkdir("/content/processed_data/train_label_data")
os.mkdir("/content/processed_data/train_label_edge_data")
os.mkdir("/content/processed_data/test_data")
os.mkdir("/content/processed_data/test_label_data")
os.mkdir("/content/processed_data/test_label_edge_data")
```

Preperation for train dataset

```
import cv2
import matplotlib

#Setting the height and width
height=224
width=320
n_classes=7

def prepare_image_data(path,data_kind):
    """Function to prepare the image dataset and save it to the location"""
    name = path.split('/')[-1]
    data_kind = data_kind
    src=path
    img = cv2.imread(src)
    img=cv2.resize(img,(width,height))
    img = np.float32(img)
    cv2.imwrite("/content/processed_data/{0}/{1}".format(data_kind,name),img)

def prepare_label_data(path,data_kind):
    """Function to prepare the image label dataset and save it to the location"""
    name = path.split('/')[-1]
    data_kind = data_kind
    label = np.zeros((height, width, n_classes))
    src=path
    img = cv2.imread(src)
    img=cv2.resize(img,(width,height))
    img1=img[:,:,0]
    for i in range(n_classes):
        label[:,:,i] = (img1==i).astype(int)
    cv2.imwrite("/content/processed_data/{0}/{1}".format(data_kind,name),np.argmax(label,axis=-1))

def prepare_edge_data(path,data_kind):
    """Function to prepare the magnitude of image lable dataset and save it to the location"""
    name = path.split('/')[-1]
    data_kind = data_kind
    label = np.zeros((height, width, n_classes))
    src=path
    img = cv2.imread(src)
    img=cv2.resize(img,(width,height))
    img1=img[:,:,0]
    img1[img1==255]=0
    gray_mask=cv2.resize(img1,(320,224))
    gX_mask = cv2.Sobel(gray_mask, cv2.CV_64F, 1, 0)
    gY_mask = cv2.Sobel(gray_mask, cv2.CV_64F, 0, 1)
    magnitude_mask = np.sqrt((gX_mask ** 2) + (gY_mask ** 2))
    magnitude_mask[magnitude_mask>0] = 1
    cv2.imwrite("/content/processed_data/{0}/{1}".format(data_kind,name),magnitude_mask)



#Creating list for storing the path of the datasets for training dataset
X_train=[]
y_train=[]
y_train_edge=[]

data_root = "/content/processed_data"

for i in range(len(train_data['image'])):
    prepare_image_data(train_data['image'][i],'train_data')
    X_train.append(os.path.join(os.path.join(data_root,'train_data'),train_data["image"][i].split('/')[-1]))

for i in range(len(train_data['mask'])):
    prepare_label_data(train_data['mask'][i],'train_label_data')
    y_train.append(os.path.join(os.path.join(data_root,'train_label_data'),train_data["mask"][i].split('/')[-1]))

for i in range(len(train_data['mask'])):
    prepare_edge_data(train_data['mask'][i],'train_label_edge_data')
    y_train_edge.append(os.path.join(os.path.join(data_root,'train_label_edge_data'),train_data["mask"][i].split('/')[-1]))



#Converting it to pandas Dataframe
X_train=pd.DataFrame(X_train)
y_train=pd.DataFrame(y_train)
y_train_edge=pd.DataFrame(y_train_edge)
```

```
#Image Data Generator
from keras.preprocessing.image import ImageDataGenerator

#Data Augmentation
datagen = ImageDataGenerator(rotation_range=30,width_shift_range=0.2, height_shift_range=0.2,horizontal_flip=True,zoom_range=0
# prepare iterator
trainX_gen = datagen.flow_from_dataframe(X_train,seed=1234,batch_size=4,class_mode=None,x_col=0,color_mode='rgb',target_size=(
trainY_gen = datagen.flow_from_dataframe(y_train,seed=1234,batch_size=4,class_mode=None,x_col=0,color_mode='grayscale',target_
trainY_gen_ed = datagen.flow_from_dataframe(y_train_edge,seed=1234,batch_size=4,class_mode=None,x_col=0,color_mode='grayscale'

train_generator = zip(trainX_gen, zip(trainY_gen,trainY_gen_ed))
```

```
Found 1403 validated image filenames.
Found 1403 validated image filenames.
Found 1403 validated image filenames.
Found 204 validated image filenames.
Found 204 validated image filenames.
Found 204 validated image filenames.
```

Preperation for validation dataset

```
X_val=[]
y_val=[]
y_val_edge=[]

for i in range(len(test_data['image'])):
    prepare_image_data(test_data['image'][i],'test_data')
    X_val.append(os.path.join(os.path.join(data_root,'test_data'),test_data["image"][i].split('/')[-1]))

for i in range(len(test_data['mask'])):
    prepare_label_data(test_data['mask'][i],'test_label_data')
    y_val.append(os.path.join(os.path.join(data_root,'test_label_data'),test_data["mask"][i].split('/')[-1]))

for i in range(len(test_data['mask'])):
    prepare_edge_data(test_data['mask'][i],'test_label_edge_data')
    y_val_edge.append(os.path.join(os.path.join(data_root,'test_label_edge_data'),test_data["mask"][i].split('/')[-1]))


#Instead of using Image datagen we will be using the complete image, label. and label edge loaded in-memory.
X_val_test=[]
y_val_test=[]
y_val_edge_test=[]

def prepare_val_image_data(path):
    src=path
    img = plt.imread(src)
    img = np.float32(img)
    return img

def prepare_val_label_data(path):
    label = np.zeros((height, width, n_classes))
    src=path
    img = cv2.imread(src)
    img=cv2.resize(img,(width,height))
    img1=img[:,:,0]
    for i in range(n_classes):
        label[:,:,i] = (img1==i).astype(int)
    return np.argmax(label,axis=-1).astype(np.float64)[:,:,np.newaxis]

def prepare_val_edge_data(path):
    src=path
    img = cv2.imread(src)
    return img[:,:,0].astype(np.float64)[:,:,np.newaxis]

for i in range(len(X_val[0])):
    X_val_test.append(prepare_val_image_data(X_val[0][i]))

for i in range(len(y_val[0])):
    y_val_test.append(prepare_val_label_data(y_val[0][i]))

for i in range(len(y_val_edge[0])):
    y_val_edge_test.append(prepare_val_edge_data(y_val_edge[0][i]))

X_val_test=np.array(X_val_test)
y_val_test=np.array(y_val_test)
y_val_edge_test=np.array(y_val_edge_test)
```

```
print(X_val_test.shape)
print(y_val_test.shape)
print(y_val_edge_test.shape)
```

```
(204, 224, 320, 3)
(204, 224, 320, 1)
(204, 224, 320, 1)
```

3. Setting Metrics

Here we will be setting with miou- mean Intersection Over Union

```
!pip install git+https://github.com/qubvel/segmentation_models
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting git+https://github.com/qubvel/segmentation_models
  Cloning https://github.com/qubvel/segmentation_models to /tmp/pip-req-build-j4wve8xv
  Running command git clone --filter=blob:none --quiet https://github.com/qubvel/segmentation_models /tmp/pip-req-build-j4wve8xv
  Resolved https://github.com/qubvel/segmentation_models to commit e951c6747f75fa9e7240816d1c79dd2e66813123
  Running command git submodule update --init --recursive -q
  Preparing metadata (setup.py) ... done
Collecting keras_applications<=1.0.8,>=1.0.7 (from segmentation-models==1.0.1)
  Downloading Keras_Applications-1.0.8-py3-none-any.whl (50 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 50.7/50.7 kB 2.1 MB/s eta 0:00:00
Collecting image-classifiers==1.0.0 (from segmentation-models==1.0.1)
  Downloading image_classifiers-1.0.0-py3-none-any.whl (19 kB)
Collecting efficientnet==1.0.0 (from segmentation-models==1.0.1)
  Downloading efficientnet-1.0.0-py3-none-any.whl (17 kB)
Requirement already satisfied: scikit-image in /usr/local/lib/python3.10/dist-packages (from efficientnet==1.0.0->segmentation-model
Requirement already satisfied: numpy>=1.9.1 in /usr/local/lib/python3.10/dist-packages (from keras_applications<=1.0.8,>=1.0.7->segm
Requirement already satisfied: h5py in /usr/local/lib/python3.10/dist-packages (from keras_applications<=1.0.8,>=1.0.7->segmentation
Requirement already satisfied: scipy>=1.4.1 in /usr/local/lib/python3.10/dist-packages (from scikit-image->efficientnet==1.0.0->segm
Requirement already satisfied: networkx>=2.2 in /usr/local/lib/python3.10/dist-packages (from scikit-image->efficientnet==1.0.0->seg
Requirement already satisfied: pillow!=7.1.0,!=7.1.1,!=8.3.0,>=6.1.0 in /usr/local/lib/python3.10/dist-packages (from scikit-image->
Requirement already satisfied: imageio>=2.4.1 in /usr/local/lib/python3.10/dist-packages (from scikit-image->efficientnet==1.0.0->se
Requirement already satisfied: tifffile>=2019.7.26 in /usr/local/lib/python3.10/dist-packages (from scikit-image->efficientnet==1.0
Requirement already satisfied: PyWavelets>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from scikit-image->efficientnet==1.0.0
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from scikit-image->efficientnet==1.0.0->s
Building wheels for collected packages: segmentation-models
  Building wheel for segmentation-models (setup.py) ... done
  Created wheel for segmentation-models: filename=segmentation_models-1.0.1-py3-none-any.whl size=33791 sha256=35e1c306b53c4c3c55e8a
  Stored in directory: /tmp/pip-ephem-wheel-cache-ggay9_w8/wheels/ce/d6/f1/5d00e82b3893c5f1ffee43bf7b8877148af09c7c9c6c4882c9
Successfully built segmentation-models
Installing collected packages: keras_applications, image-classifiers, efficientnet, segmentation-models
Successfully installed efficientnet-1.0.0 image-classifiers-1.0.0 keras_applications-1.0.8 segmentation-models-1.0.1
```

```
import tensorflow as tf
tf.compat.v1.enable_eager_execution()
from tensorflow import keras
from tensorflow.keras.layers import *
from tensorflow.keras.preprocessing import image
from tensorflow.keras.models import Model, load_model
from tensorflow.keras.layers import UpSampling2D
from tensorflow.keras.layers import MaxPooling2D, GlobalAveragePooling2D
from tensorflow.keras.layers import concatenate
from tensorflow.keras.layers import Multiply
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint
from tensorflow.keras import backend as K
from tensorflow.keras.layers import Input, Add, Dense, Activation, ZeroPadding2D, BatchNormalization, Flatten, Conv2D, AverageP
from tensorflow.keras.regularizers import L2
from tensorflow.keras.models import Model, load_model
from tensorflow.keras.utils import plot_model
from tensorflow.keras.initializers import glorot_uniform
K.set_image_data_format('channels_last')
K.set_learning_phase(1)
```

```
/usr/local/lib/python3.10/dist-packages/keras/backend.py:452: UserWarning: `tf.keras.backend.set_learning_phase` is deprecated and w
  warnings.warn(
```

```
#Settting steps per epoch
steps_per_epoch=X_train.shape[0]//2
print(steps_per_epoch)
validation_steps=X_val.shape[0]//2
print(validation_steps)
```

```
701
102
```

```
def IoU(y_val, y_pred):
    class_iou = []
    n_classes = 7

    y_predi = np.argmax(y_pred, axis=-1)
    y_truei = y_val[:,:,:,0]

    for c in range(n_classes):
        TP = np.sum((y_truei == c) & (y_predi == c))
        FP = np.sum((y_truei != c) & (y_predi == c))
        FN = np.sum((y_truei == c) & (y_predi != c))
        IoU = TP / float(TP + FP + FN)
        if(float(TP + FP + FN) == 0):
          IoU=TP/0.001
        class_iou.append(IoU)
    MIoU=sum(class_iou)/n_classes
    return MIoU
def miou( y_true, y_pred ) :
    score = tf.py_function( lambda y_true, y_pred : IoU( y_true, y_pred).astype('float32'),
                        [y_true, y_pred],
                        'float32')
    return score
```

Creating Architecture

```
!pip install git+https://github.com/qubvel/segmentation_models
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting git+https://github.com/qubvel/segmentation_models
  Cloning https://github.com/qubvel/segmentation_models to /tmp/pip-req-build-8zfy8n73
  Running command git clone --filter=blob:none --quiet https://github.com/qubvel/segmentation_models /tmp/pip-req-build-8zfy8n73
  Resolved https://github.com/qubvel/segmentation_models to commit e951c6747f75fa9e7240816d1c79dd2e66813123
  Running command git submodule update --init --recursive -q
  Preparing metadata (setup.py) ... done
Collecting keras_applications<=1.0.8,>=1.0.7 (from segmentation-models==1.0.1)
  Downloading Keras_Applications-1.0.8-py3-none-any.whl (50 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 50.7/50.7 kB 2.7 MB/s eta 0:00:00
Collecting image-classifiers==1.0.0 (from segmentation-models==1.0.1)
  Downloading image_classifiers-1.0.0-py3-none-any.whl (19 kB)
Collecting efficientnet==1.0.0 (from segmentation-models==1.0.1)
  Downloading efficientnet-1.0.0-py3-none-any.whl (17 kB)
Requirement already satisfied: scikit-image in /usr/local/lib/python3.10/dist-packages (from efficientnet==1.0.0->segmentation-model
Requirement already satisfied: numpy>=1.9.1 in /usr/local/lib/python3.10/dist-packages (from keras_applications<=1.0.8,>=1.0.7->segm
Requirement already satisfied: h5py in /usr/local/lib/python3.10/dist-packages (from keras_applications<=1.0.8,>=1.0.7->segmentation
Requirement already satisfied: scipy>=1.4.1 in /usr/local/lib/python3.10/dist-packages (from scikit-image->efficientnet==1.0.0->segm
Requirement already satisfied: networkx>=2.2 in /usr/local/lib/python3.10/dist-packages (from scikit-image->efficientnet==1.0.0->seg
Requirement already satisfied: pillow!=7.1.0,!=7.1.1,!=8.3.0,>=6.1.0 in /usr/local/lib/python3.10/dist-packages (from scikit-image->
Requirement already satisfied: imageio>=2.4.1 in /usr/local/lib/python3.10/dist-packages (from scikit-image->efficientnet==1.0.0->se
Requirement already satisfied: tifffile>=2019.7.26 in /usr/local/lib/python3.10/dist-packages (from scikit-image->efficientnet==1.0
Requirement already satisfied: PyWavelets>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from scikit-image->efficientnet==1.0.0
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from scikit-image->efficientnet==1.0.0->s
Building wheels for collected packages: segmentation-models
  Building wheel for segmentation-models (setup.py) ... done
  Created wheel for segmentation-models: filename=segmentation_models-1.0.1-py3-none-any.whl size=33791 sha256=d3ed3f2c5fba5f2ff643
  Stored in directory: /tmp/pip-ephem-wheel-cache-fayrwfxr/wheels/ce/d6/f1/5d00e82b3893c5f1ffee43bf7b8877148af09c7c9c6c4882c9
Successfully built segmentation-models
Installing collected packages: keras_applications, image-classifiers, efficientnet, segmentation-models
Successfully installed efficientnet-1.0.0 image-classifiers-1.0.0 keras_applications-1.0.8 segmentation-models-1.0.1
```

```
!pip install imgaug
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: imgaug in /usr/local/lib/python3.10/dist-packages (0.4.0)
Requirement already satisfied: six in /usr/local/lib/python3.10/dist-packages (from imgaug) (1.16.0)
Requirement already satisfied: numpy>=1.15 in /usr/local/lib/python3.10/dist-packages (from imgaug) (1.22.4)
Requirement already satisfied: scipy in /usr/local/lib/python3.10/dist-packages (from imgaug) (1.10.1)
Requirement already satisfied: Pillow in /usr/local/lib/python3.10/dist-packages (from imgaug) (8.4.0)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist-packages (from imgaug) (3.7.1)
Requirement already satisfied: scikit-image>=0.14.2 in /usr/local/lib/python3.10/dist-packages (from imgaug) (0.19.3)
Requirement already satisfied: opencv-python in /usr/local/lib/python3.10/dist-packages (from imgaug) (4.7.0.72)
Requirement already satisfied: imageio in /usr/local/lib/python3.10/dist-packages (from imgaug) (2.25.1)
Requirement already satisfied: Shapely in /usr/local/lib/python3.10/dist-packages (from imgaug) (2.0.1)
Requirement already satisfied: networkx>=2.2 in /usr/local/lib/python3.10/dist-packages (from scikit-image>=0.14.2->imgaug) (3.1)
Requirement already satisfied: tifffile>=2019.7.26 in /usr/local/lib/python3.10/dist-packages (from scikit-image>=0.14.2->imgaug) (2
Requirement already satisfied: PyWavelets>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from scikit-image>=0.14.2->imgaug) (1.4
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from scikit-image>=0.14.2->imgaug) (23.1
```

```
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib->imgaug) (1.0.7)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib->imgaug) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib->imgaug) (4.39.3)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib->imgaug) (1.4.4)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib->imgaug) (3.0.9)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.10/dist-packages (from matplotlib->imgaug) (2.8.2)
```

```
import tensorflow as tf
tf.compat.v1.enable_eager_execution()
from tensorflow import keras
from tensorflow.keras.layers import *
from tensorflow.keras.preprocessing import image
from tensorflow.keras.models import Model, load_model
from tensorflow.keras.layers import UpSampling2D
from tensorflow.keras.layers import MaxPooling2D, GlobalAveragePooling2D
from tensorflow.keras.layers import concatenate
from tensorflow.keras.layers import Multiply
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint
from tensorflow.keras import backend as K
from tensorflow.keras.layers import Input, Add, Dense, Activation, ZeroPadding2D, BatchNormalization, Flatten, Conv2D, AveragePooling2D
from tensorflow.keras.regularizers import L2
from tensorflow.keras.models import Model, load_model
from tensorflow.keras.utils import plot_model
from tensorflow.keras.initializers import glorot_uniform
K.set_image_data_format('channels_last')
K.set_learning_phase(1)
```

```
!pip install -U -q segmentation-models
!pip install -q tensorflow==2.2.1
!pip install -q keras==2.5
import os
os.environ["SM_FRAMEWORK"] = "tf.keras"
```

```
ERROR: Could not find a version that satisfies the requirement tensorflow==2.2.1 (from versions: 2.8.0rc0, 2.8.0rc1, 2.8.0, 2.8.1, 2
ERROR: No matching distribution found for tensorflow==2.2.1
ERROR: Could not find a version that satisfies the requirement keras==2.5 (from versions: 0.2.0, 0.3.0, 0.3.1, 0.3.2, 0.3.3, 1.0.0,
ERROR: No matching distribution found for keras==2.5
```

Decoder Block

The decoder block is a simple decoding Unet block that upsamples the input and concatenates with the skip connection outputs.

```
Input: X --> X[0],X[1]
          X[0] contains the previous layer output.
          X[1] contains the connection output.


Architecture:
    Upsampling2D(X[0])(mode="bilinear", size=(2,2))
    concatenate(upsample,X[1])
    Conv2D
    BatchNormalization
    Relu
    Conv2D
    BatchNormalization
    Relu
```

```
#Setting the Decoder Block
class DecoderBlock(tf.keras.layers.Layer):
  def __init__(self,filters,stride=1,kernel_size=3,name='dec_block'):
    super().__init__(name=name)
    self.stride=stride
    self.filters=filters
    self.kernel_size=kernel_size

  def build(self,input_shape):
    self.upsampler=UpSampling2D(size=(2,2),interpolation='bilinear')
    self.conv_1=Conv2D(filters=self.filters,kernel_size=self.kernel_size,strides=self.stride,padding='same',kernel_initializer=
    self.bn_1=BatchNormalization()
    self.relu_1=Activation('relu')
    self.conv_2=Conv2D(filters=self.filters,kernel_size=self.kernel_size,strides=self.stride,padding='same',kernel_initializer=
    self.bn_2=BatchNormalization()
    self.relu_2=Activation('relu')

  def call(self,X):
    prev_layer=X[0]
    connect_layer=X[1]
    upsample_out=self.upsampler(prev_layer)
    concat_out=concatenate([upsample_out,connect_layer])
    out_1=self.conv_1(concat_out)
    out_1=self.bn_1(out_1)
    out_1=self.relu_1(out_1)
    out_1=self.conv_2(out_1)
    out_1=self.bn_2(out_1)
    out_1=self.relu_2(out_1)
    return out_1
```

Edge Modules

GCL Block

GCL Block acts as an attention block which help in predicting the image edges.

```
Input: X --> X[0], X[1]
             X[0] is the output edge flow
             X[1] is the output from the regular decoder flow


Architecture:
    concatenate(X[0],X[1])
    Conv2D(filters=1, kernel_size=1) #This is acts as attention layer for edge detection
    Sigmoid
    Elementwise multiplication(X[0],Conv2D(attention))
    Elementwise addition(elem_multi, X[0])
    Conv2D(filters=1, kernel_size=1)
```

Residual Block

This block is simple residual block that does residual connection.

```
Input: X --> X
             X is the output of previous layer

Architecture:
    Conv2D(filters,kernel_size=1)
    BatchNormalization
    Relu
    Conv2D(filters*2,kernel_size=kernel_size)
    BatchNormalization
    Relu
    Conv2D(filters,kernel_size=1)
    BatchNormalization
    Relu
```

Upsampler Block

This block upsamples the shape output

```
Input: X --> X
             X is the output of previous layer
```

```
Architecture:
    UpSampling2D(mode="bilinear",size=(2,2))
    Conv2D(filters,kernel_size=kernel_size)
    BatchNormalization
    Relu
    Conv2D(filters*2,kernel_size=kernel_size)
    BatchNormalization
    Relu
    Conv2D(filters,kernel_size=kernel_size)
    BatchNormalization
    Relu
```

```
#Setting the GCL Block
class GclBlock(tf.keras.layers.Layer):
    def __init__(self,kernel=3,filters=1,stride=1,pool_size=None,name='gcl',padding='valid'):
      super().__init__(name=name)
      self.kernel=kernel
      self.filters=filters
      self.stride=stride
      self.pool_size=pool_size
      self.padding=padding
      if self.pool_size:
        self.padding='same'
      if self.stride>1:
        self.padding='valid'

    def build(self,input_shape):
      self.att_layer1=Conv2D(filters=1,kernel_size=1,kernel_initializer="he_normal")
      self.bn_1_layer=BatchNormalization()
      self.final_layer=Conv2D(filters=1,kernel_size=1,strides=self.stride,kernel_initializer="he_normal")
      self.relu_act=Activation('relu')
      self.bn_1_layer=BatchNormalization()
      self.activation_sig=Activation('sigmoid')
      self.elem_multiply=Multiply()
      self.elem_addition=Add()
      if self.pool_size:
        self.pool_layer=MaxPooling2D((self.pool_size,self.pool_size))
    def call(self,X):
      shape_flow,regular_flow=X[0],X[1]
      concat_out=concatenate([shape_flow,regular_flow],axis=-1)
      conv_out=self.att_layer1(concat_out)
      att_out=self.activation_sig(conv_out)
      multiply_out=self.elem_multiply([shape_flow,att_out])
      add_out=self.elem_addition([multiply_out,shape_flow])
      final_out=self.final_layer(add_out)
      if self.pool_size:
        final_out=self.pool_layer(final_out)
      return final_out

#Setting Residual Block
class ResBlock(tf.keras.layers.Layer):
    def __init__(self,kernel=3,filters=1,name='res_block'):
      super().__init__(name=name)
      self.kernel=kernel
      self.filters=filters
    def build(self,input_shape):
      self.conv1=Conv2D(filters=self.filters,kernel_size=1,padding='same')
      self.conv2=Conv2D(filters=self.filters*2,kernel_size=self.kernel,padding='same')
      self.conv3=Conv2D(filters=self.filters,kernel_size=1,padding='same')
      self.bn1=BatchNormalization()
      self.bn2=BatchNormalization()
      self.bn3=BatchNormalization()
      self.activation1=Activation('relu')
      self.activation2=Activation('relu')
      self.activation3=Activation('relu')
      self.elem_addition=Add()
    def call(self,X):
      conv1_out=self.conv1(X)
      bn1_out=self.bn1(conv1_out)
      act1_out=self.activation1(bn1_out)
      conv2_out=self.conv2(act1_out)
      bn2_out=self.bn2(conv2_out)
      act2_out=self.activation2(bn2_out)
      conv3_out=self.conv3(act2_out)
      bn3_out=self.bn3(conv3_out)
      add_out=self.elem_addition([bn3_out,X])
      final_out=self.activation3(add_out)
      return final_out

#Setting Upsampler Block
class UpsamplerBlock(tf.keras.layers.Layer):
  def __init__(self,filters,upsample_size,name="upsamp_block",kernel_size=3,strides=1):
    super().__init__(name=name)
    self.filters=filters
    self.kernel_size=kernel_size
    self.strides=strides
    self.upsample_size=upsample_size

  def build(self,input_shape):
    self.upsampler=UpSampling2D(size=(self.upsample_size,self.upsample_size),interpolation="bilinear")
```

```
    self.conv_out_1=Conv2D(filters=self.filters,kernel_size=self.kernel_size,strides=self.strides,padding='same',kernel_initia
    self.batch_norm_1=BatchNormalization()
    self.act_1=Activation('relu')
    self.conv_out_2=Conv2D(filters=self.filters,kernel_size=self.kernel_size,strides=self.strides,padding='same',kernel_initia
    self.batch_norm_2=BatchNormalization()
    self.act_2=Activation('relu')
    self.conv_out_3=Conv2D(filters=self.filters,kernel_size=1,strides=self.strides,padding='same',kernel_initializer="he_norma

  def call(self,X):
    output=self.upsampler(X)
    output=self.conv_out_1(output)
    output=self.batch_norm_1(output)
    output=self.act_1(output)
    output=self.conv_out_2(output)
    output=self.batch_norm_2(output)
    output=self.act_2(output)
    output=self.conv_out_3(output)
    return output
```

Training Plain U-net with ResNet-50 as encoder

```
import segmentation_models
model = segmentation_models.Unet(backbone_name='resnet50', encoder_weights='imagenet', encoder_freeze=False,input_shape=(224,3
model.summary()

import segmentation_models as sm
optim=tf.keras.optimizers.Adam(0.001)

dice_loss=sm.losses.DiceLoss()
bin_ce=tf.keras.losses.BinaryCrossentropy()
categorical_ce=tf.keras.losses.SparseCategoricalCrossentropy()
biou=sm.metrics.IOUScore()
binary_loss=sm.losses.BinaryFocalLoss()
model.compile(optim,loss={"softmax":lambda y_true,y_pred: dice_loss(y_true,y_pred)+categorical_ce(y_true,y_pred)},
                  metrics={"softmax":miou})

callbacks = [
    tf.keras.callbacks.ModelCheckpoint('best_model_canet_t3.h5', save_weights_only=True, save_best_only=True, mode='min'),
    tf.keras.callbacks.TensorBoard('TBlog_canet_t3',histogram_freq=1),
#    tf.keras.callbacks.EarlyStopping(monitor='val_loss',min_delta=0.01,patience=3,verbose=1,mode='min'),
        tf.keras.callbacks.ReduceLROnPlateau(monitor='val_loss',factor=0.5,patience=4,verbose=1,mode='min',min_lr=0.00001,min_
]
history_resnet_1 = model.fit(train_generator, epochs=40,steps_per_epoch=steps_per_epoch,validation_data=(X_val_test,y_val_test
```

```
701/701 [==============================] - 172s 246ms/step - loss: 0.9315 - miou: 0.7366 - val_loss: 1.0748 - val_miou: 0.6620 -
Epoch 34/40
701/701 [==============================] - ETA: 0s - loss: 0.9298 - miou: 0.7390
Epoch 34: ReduceLROnPlateau reducing learning rate to 0.0001250000059371814.
701/701 [==============================] - 172s 246ms/step - loss: 0.9298 - miou: 0.7390 - val_loss: 1.0719 - val_miou: 0.6617 -
Epoch 35/40
701/701 [==============================] - 172s 246ms/step - loss: 0.9244 - miou: 0.7448 - val_loss: 1.0718 - val_miou: 0.6644 -
Epoch 36/40
701/701 [==============================] - 173s 246ms/step - loss: 0.9229 - miou: 0.7455 - val_loss: 1.0703 - val_miou: 0.6667 -
Epoch 37/40
701/701 [==============================] - 172s 246ms/step - loss: 0.9213 - miou: 0.7469 - val_loss: 1.0717 - val_miou: 0.6653 -
Epoch 38/40
701/701 [==============================] - ETA: 0s - loss: 0.9198 - miou: 0.7488
Epoch 38: ReduceLROnPlateau reducing learning rate to 6.25000029685907e-05.
701/701 [==============================] - 172s 245ms/step - loss: 0.9198 - miou: 0.7488 - val_loss: 1.0748 - val_miou: 0.6636 -
Epoch 39/40
701/701 [==============================] - 173s 246ms/step - loss: 0.9171 - miou: 0.7509 - val_loss: 1.0696 - val_miou: 0.6677 -
Epoch 40/40
701/701 [==============================] - 175s 249ms/step - loss: 0.9177 - miou: 0.7516 - val_loss: 1.0748 - val_miou: 0.6678 -
```

```
#Prediction on validation dataset
pred = []
for i in range(len(X_val_test)):
  pred.append(model.predict(X_val_test[i][np.newaxis,:,:,:])[0])
pred1 = np.array(pred)
```

```
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 26ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 26ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 27ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 26ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 27ms/step
1/1 [==============================] - 0s 27ms/step
1/1 [==============================] - 0s 26ms/step
1/1 [==============================] - 0s 27ms/step
1/1 [==============================] - 0s 27ms/step
1/1 [==============================] - 0s 32ms/step
1/1 [==============================] - 0s 28ms/step
1/1 [==============================] - 0s 28ms/step
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 27ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 35ms/step
```

```
#Claculating class wise mIoU for the validation dataset
def class_wise_IoU(y_val, y_pred):
    class_iou = []
    n_classes = 7

    y_predi = np.argmax(y_pred, axis=-1)
    y_truei = y_val[:,:,:,0]

    for c in range(n_classes):
        TP = np.sum((y_truei == c) & (y_predi == c))
        FP = np.sum((y_truei != c) & (y_predi == c))
        FN = np.sum((y_truei == c) & (y_predi != c))
        IoU = TP / float(TP + FP + FN)
        if(float(TP + FP + FN) == 0):
          IoU=TP/0.001
        class_iou.append(IoU)
    return class_iou
cwiou = class_wise_IoU(y_val_test,pred1)
print("overall IoU score for resnet50 with Unet: {}".format(IoU(y_val_test,pred1)))
print("class wise IoU score for resnet50 with Unet")
for i in range(len(cwiou)):
    print("class {0} IoU score: {1}".format(i,cwiou[i]))
```

```
overall IoU score for resnet50 with Unet: 0.6786559541382934
class wise IoU score for resnet50 with Unet
class 0 IoU score: 0.9353041127077119
class 1 IoU score: 0.38891453537497683
class 2 IoU score: 0.4604198143751349
class 3 IoU score: 0.7635513712545677
class 4 IoU score: 0.4819599950720166
class 5 IoU score: 0.7625738189882885
class 6 IoU score: 0.9578680311953571
```

Training Modified U-Net-Edge Model with resnet-50 encoder

```
resnet=tf.keras.applications.ResNet50(include_top=False,weights='imagenet',input_shape=(224,320,3))
resnet.summary()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/resnet/resnet50_weights_tf_dim_ordering_tf_ker
94765736/94765736 [==============================] - 6s 0us/step
Model: "resnet50"
__________________________________________________________________________________________________
 Layer (type)                   Output Shape         Param #     Connected to
==================================================================================================
 input_1 (InputLayer)           [(None, 224, 320, 3  0           []
                                )]

 conv1_pad (ZeroPadding2D)      (None, 230, 326, 3)  0           ['input_1[0][0]']

 conv1_conv (Conv2D)            (None, 112, 160, 64  9472        ['conv1_pad[0][0]']
                                )

 conv1_bn (BatchNormalization)  (None, 112, 160, 64  256         ['conv1_conv[0][0]']
                                )

 conv1_relu (Activation)        (None, 112, 160, 64  0           ['conv1_bn[0][0]']
                                )

 pool1_pad (ZeroPadding2D)      (None, 114, 162, 64  0           ['conv1_relu[0][0]']
                                )

 pool1_pool (MaxPooling2D)      (None, 56, 80, 64)   0           ['pool1_pad[0][0]']

 conv2_block1_1_conv (Conv2D)   (None, 56, 80, 64)   4160        ['pool1_pool[0][0]']

 conv2_block1_1_bn (BatchNormal  (None, 56, 80, 64)  256         ['conv2_block1_1_conv[0][0]']
 ization)

 conv2_block1_1_relu (Activatio  (None, 56, 80, 64)  0           ['conv2_block1_1_bn[0][0]']
 n)

 conv2_block1_2_conv (Conv2D)   (None, 56, 80, 64)   36928       ['conv2_block1_1_relu[0][0]']

 conv2_block1_2_bn (BatchNormal  (None, 56, 80, 64)  256         ['conv2_block1_2_conv[0][0]']
 ization)

 conv2_block1_2_relu (Activatio  (None, 56, 80, 64)  0           ['conv2_block1_2_bn[0][0]']
 n)

 conv2_block1_0_conv (Conv2D)   (None, 56, 80, 256)  16640       ['pool1_pool[0][0]']

 conv2_block1_3_conv (Conv2D)   (None, 56, 80, 256)  16640       ['conv2_block1_2_relu[0][0]']

 conv2_block1_0_bn (BatchNormal  (None, 56, 80, 256)  1024       ['conv2_block1_0_conv[0][0]']
```

```
ization)

conv2_block1_3_bn (BatchNormal  (None, 56, 80, 256)  1024       ['conv2_block1_3_conv[0][0]']
ization)

conv2_block1_add (Add)         (None, 56, 80, 256)  0           ['conv2_block1_0_bn[0][0]',
                                                                 'conv2_block1_3_bn[0][0]']

conv2_block1_out (Activation)  (None, 56, 80, 256)  0           ['conv2_block1_add[0][0]']

conv2_block2_1_conv (Conv2D)   (None, 56, 80, 64)   16448       ['conv2_block1_out[0][0]']
```

```
def encoder(inputs):
    skip_connections = []

    model = tf.keras.applications.ResNet50(include_top=False, weights='imagenet', input_tensor=inputs)
    names = ["conv1_relu", "conv2_block3_out", "conv3_block4_out", "conv4_block6_out"]
    for layer in model.layers:
        layer.trainable=False
    for name in names:
        skip_connections.append(model.get_layer(name).output)

    output = model.get_layer("conv5_block3_out").output
    return output, skip_connections

keras.backend.clear_session()
X_inp = Input(shape=(224,320,3))
# Encoder
c_5_5,conv_blocks = encoder(X_inp)
c_1_1,c_2_2,c_3_3,c_4_4=conv_blocks[0],conv_blocks[1],conv_blocks[2],conv_blocks[3]

# Decoder
dec_1_out=DecoderBlock(filters=256,name="decoder_1")((c_5_5,c_4_4))
dec_2_out=DecoderBlock(filters=128,name="decoder_2")((dec_1_out,c_3_3))
dec_3_out=DecoderBlock(filters=64,name="decoder_3")((dec_2_out,c_2_2))
dec_4_out=DecoderBlock(filters=32,name="decoder_4")((dec_3_out,c_1_1))
dec_5_upsamp=UpSampling2D(size=(2,2),interpolation='bilinear')(dec_4_out)
dec_5_conv1=Conv2D(filters=16,name="decoder_5_conv1",strides=1,kernel_size=3,padding="same")(dec_5_upsamp)
dec_5_bn1=BatchNormalization()(dec_5_conv1)
dec_5_act1=Activation("relu")(dec_5_bn1)
dec_5_conv2=Conv2D(filters=16,name="decoder_6_conv2",strides=1,kernel_size=3,padding="same")(dec_5_act1)
dec_5_bn2=BatchNormalization()(dec_5_conv2)
dec_5_act2=Activation("relu")(dec_5_bn2)

#Edge
conv_shape_out_1=Conv2D(filters=128,kernel_size=3,strides=1,padding="same")(dec_2_out)
upsample_out_1=UpsamplerBlock(filters=128,upsample_size=2,name="upsampler_1")(conv_shape_out_1)
res_out_1=ResBlock(filters=128,name="res_1")(upsample_out_1)
gcl_out_1=GclBlock(name="gcl_1")([res_out_1,dec_3_out])
upsample_out_2=UpsamplerBlock(filters=1,upsample_size=2,name="upsampler_2")(gcl_out_1)
res_out_2=ResBlock(filters=1,name="res_2")(upsample_out_2)
gcl_out_2=GclBlock(name="gcl_2")([res_out_2,dec_4_out])
upsample_out_3=UpsamplerBlock(filters=1,upsample_size=2,name="upsampler_3")(gcl_out_2)
res_out_3=ResBlock(filters=1,name="res_3")(upsample_out_3)
gcl_out_3=GclBlock(name='gcl_3')([res_out_3,dec_5_act2])
conv_shape_out_2=Conv2D(filters=1,kernel_size=3,strides=1,padding='same')(gcl_out_3)
sigmoid_out=Activation("sigmoid",name="sigmoid_out")(conv_shape_out_2)

#Final concatenation layers
concat_final_out=concatenate([sigmoid_out,dec_5_act2])
final_conv=Conv2D(filters=7,strides=1,kernel_size=3,padding="same")(concat_final_out)
softmax_out=Activation('softmax',name="softmax_out")(final_conv)

model_resnet=Model(inputs=X_inp,outputs=[softmax_out,sigmoid_out])

for layer in model_resnet.layers:
  layer.trainable=True
model_resnet.summary()

#Optimizer is Adam with lr 0.001
import segmentation_models as sm
optim=tf.keras.optimizers.Adam(0.001)

#For the final softmax, loss will be diceloss + categorical cross entropy
#For the Edge sigmoid, loss will be diceloss + binary cross entropy
#For the final softmax. metrics will be miou
#For the edge sigmoid. metrics will be biou(simple IoU score)

dice_loss=sm.losses.DiceLoss()
bin_ce=tf.keras.losses.BinaryCrossentropy()
categorical_ce=tf.keras.losses.SparseCategoricalCrossentropy()
biou=sm.metrics.IOUScore()
model_resnet.compile(optim,loss={"softmax_out":lambda y_true,y_pred: dice_loss(y_true,y_pred)+categorical_ce(y_true,y_pred),
                                 "sigmoid_out":lambda y_true,y_pred: dice_loss(y_true,y_pred)+bin_ce(y_true,y_pred)},
                     metrics={"softmax_out":miou,"sigmoid_out":biou})

callbacks = [
    tf.keras.callbacks.ModelCheckpoint('best_model_canet_t3.h5', save_weights_only=True, save_best_only=True, mode='min'),
    tf.keras.callbacks.TensorBoard('TBlog_canet_t3',histogram_freq=1),
        tf.keras.callbacks.ReduceLROnPlateau(monitor='val_loss',factor=0.5,patience=4,verbose=1,mode='min',min_lr=0.00001,min_
```

```
]
history_resnet_1 = model_resnet.fit(train_generator, epochs=40,steps_per_epoch=steps_per_epoch,validation_data=(X_val_test,(y_
```

```
t_loss: 1.0757 - val_sigmoid_out_loss: 0.6214 - val_softmax_out_miou: 0.6543 - val_sigmoid_out_iou_score: 0.4128 - lr: 5.0000e-04
t_loss: 1.0723 - val_sigmoid_out_loss: 0.6316 - val_softmax_out_miou: 0.6650 - val_sigmoid_out_iou_score: 0.3995 - lr: 5.0000e-04
t_loss: 1.0642 - val_sigmoid_out_loss: 0.6218 - val_softmax_out_miou: 0.6588 - val_sigmoid_out_iou_score: 0.4122 - lr: 5.0000e-04
t_loss: 1.0616 - val_sigmoid_out_loss: 0.6166 - val_softmax_out_miou: 0.6652 - val_sigmoid_out_iou_score: 0.4180 - lr: 5.0000e-04
t_loss: 1.0699 - val_sigmoid_out_loss: 0.6234 - val_softmax_out_miou: 0.6547 - val_sigmoid_out_iou_score: 0.4232 - lr: 5.0000e-04
t_loss: 1.0714 - val_sigmoid_out_loss: 0.6253 - val_softmax_out_miou: 0.6622 - val_sigmoid_out_iou_score: 0.4204 - lr: 5.0000e-04
t_loss: 1.0824 - val_sigmoid_out_loss: 0.6211 - val_softmax_out_miou: 0.6549 - val_sigmoid_out_iou_score: 0.4187 - lr: 5.0000e-04

t_loss: 1.0814 - val_sigmoid_out_loss: 0.6283 - val_softmax_out_miou: 0.6584 - val_sigmoid_out_iou_score: 0.4161 - lr: 5.0000e-04
t_loss: 1.0746 - val_sigmoid_out_loss: 0.6139 - val_softmax_out_miou: 0.6665 - val_sigmoid_out_iou_score: 0.4221 - lr: 2.5000e-04
t_loss: 1.0770 - val_sigmoid_out_loss: 0.6124 - val_softmax_out_miou: 0.6678 - val_sigmoid_out_iou_score: 0.4232 - lr: 2.5000e-04
t_loss: 1.0844 - val_sigmoid_out_loss: 0.6209 - val_softmax_out_miou: 0.6679 - val_sigmoid_out_iou_score: 0.4273 - lr: 2.5000e-04

t_loss: 1.0750 - val_sigmoid_out_loss: 0.6147 - val_softmax_out_miou: 0.6732 - val_sigmoid_out_iou_score: 0.4241 - lr: 2.5000e-04
t_loss: 1.0764 - val_sigmoid_out_loss: 0.6112 - val_softmax_out_miou: 0.6715 - val_sigmoid_out_iou_score: 0.4245 - lr: 1.2500e-04
t_loss: 1.0725 - val_sigmoid_out_loss: 0.6139 - val_softmax_out_miou: 0.6725 - val_sigmoid_out_iou_score: 0.4262 - lr: 1.2500e-04
t_loss: 1.0807 - val_sigmoid_out_loss: 0.6129 - val_softmax_out_miou: 0.6695 - val_sigmoid_out_iou_score: 0.4238 - lr: 1.2500e-04

t_loss: 1.0818 - val_sigmoid_out_loss: 0.6129 - val_softmax_out_miou: 0.6715 - val_sigmoid_out_iou_score: 0.4289 - lr: 1.2500e-04
t_loss: 1.0787 - val_sigmoid_out_loss: 0.6115 - val_softmax_out_miou: 0.6723 - val_sigmoid_out_iou_score: 0.4276 - lr: 6.2500e-05
t_loss: 1.0802 - val_sigmoid_out_loss: 0.6120 - val_softmax_out_miou: 0.6730 - val_sigmoid_out_iou_score: 0.4282 - lr: 6.2500e-05
t_loss: 1.0840 - val_sigmoid_out_loss: 0.6123 - val_softmax_out_miou: 0.6730 - val_sigmoid_out_iou_score: 0.4278 - lr: 6.2500e-05

t_loss: 1.0799 - val_sigmoid_out_loss: 0.6126 - val_softmax_out_miou: 0.6736 - val_sigmoid_out_iou_score: 0.4279 - lr: 6.2500e-05
t_loss: 1.0790 - val_sigmoid_out_loss: 0.6113 - val_softmax_out_miou: 0.6746 - val_sigmoid_out_iou_score: 0.4282 - lr: 3.1250e-05
t_loss: 1.0839 - val_sigmoid_out_loss: 0.6128 - val_softmax_out_miou: 0.6730 - val_sigmoid_out_iou_score: 0.4289 - lr: 3.1250e-05
t_loss: 1.0828 - val_sigmoid_out_loss: 0.6127 - val_softmax_out_miou: 0.6730 - val_sigmoid_out_iou_score: 0.4283 - lr: 3.1250e-05

t_loss: 1.0852 - val_sigmoid_out_loss: 0.6125 - val_softmax_out_miou: 0.6718 - val_sigmoid_out_iou_score: 0.4290 - lr: 3.1250e-05
```

```
#Prediction on the validation dataset
pred = []
for i in range(len(X_val_test)):
  pred.append(model_resnet.predict(X_val_test[i][np.newaxis,:,:,:])[0])
pred1 = np.array(pred)
pred1 = pred1.reshape(pred1.shape[0],pred1.shape[2],pred1.shape[3],pred1.shape[4])
```

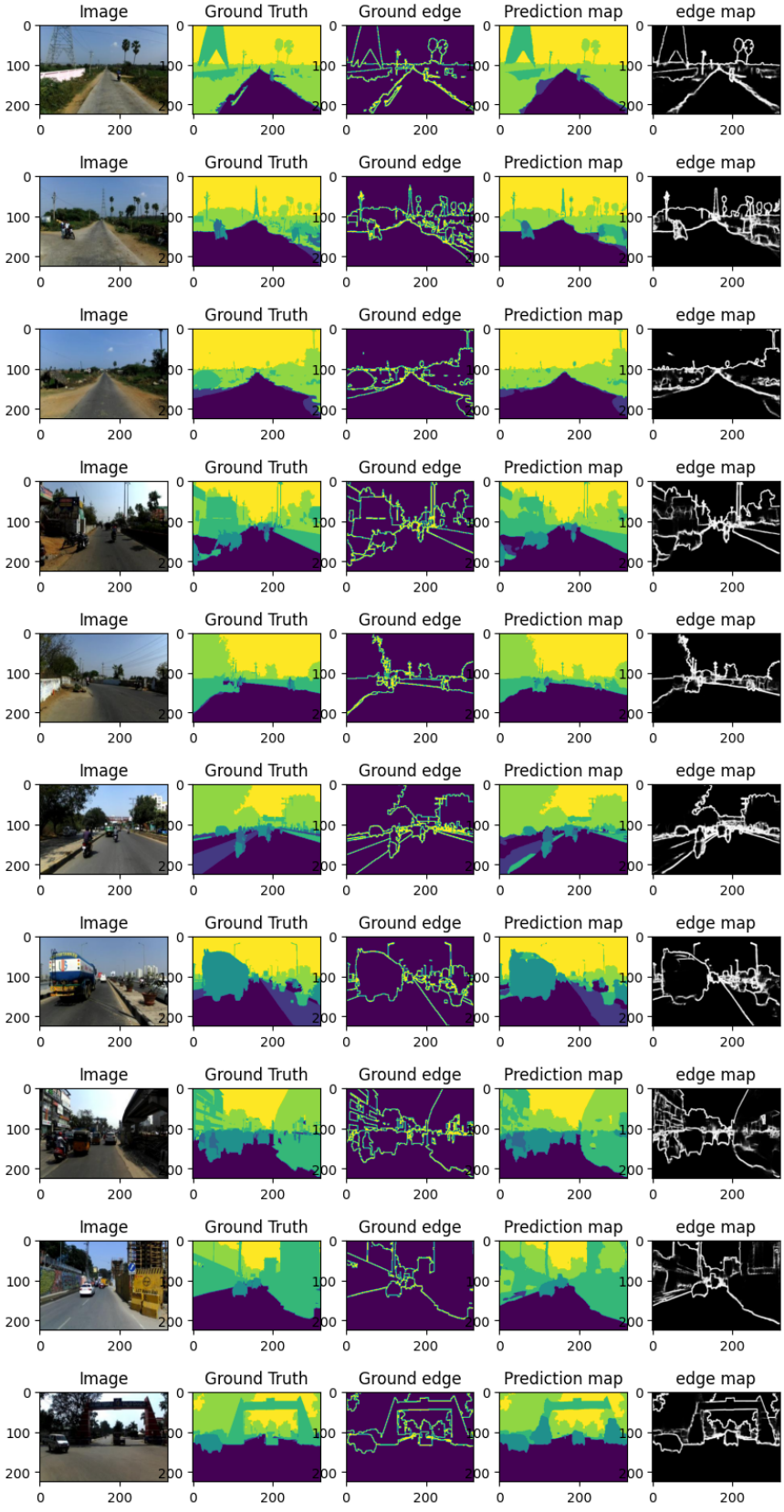
```
1/1 [==============================] - 0s 28ms/step
1/1 [==============================] - 0s 37ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 32ms/step
1/1 [==============================] - 0s 28ms/step
1/1 [==============================] - 0s 33ms/step
1/1 [==============================] - 0s 33ms/step
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 33ms/step
1/1 [==============================] - 0s 32ms/step
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 38ms/step
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 32ms/step
1/1 [==============================] - 0s 32ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 28ms/step
1/1 [==============================] - 0s 32ms/step
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 28ms/step
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 27ms/step
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 27ms/step
1/1 [==============================] - 0s 28ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 27ms/step
1/1 [==============================] - 0s 27ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 28ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 28ms/step
1/1 [==============================] - 0s 28ms/step
1/1 [==============================] - 0s 28ms/step
1/1 [==============================] - 0s 28ms/step
```

```
#Class wise IoU score on validation dataset
def class_wise_IoU(y_val, y_pred):
    class_iou = []
    n_classes = 7

    y_predi = np.argmax(y_pred, axis=-1)
    y_truei = y_val[:,:,:,0]

    for c in range(n_classes):
        TP = np.sum((y_truei == c) & (y_predi == c))
        FP = np.sum((y_truei != c) & (y_predi == c))
        FN = np.sum((y_truei == c) & (y_predi != c))
        IoU = TP / float(TP + FP + FN)
        if(float(TP + FP + FN) == 0):
          IoU=TP/0.001
        class_iou.append(IoU)
    return class_iou
cwiou = class_wise_IoU(y_val_test,pred1)
print("overall IoU score for resnet50: {}".format(IoU(y_val_test,pred1)))
print("class wise IoU score for resnet50")
for i in range(len(cwiou)):
    print("class {0} IoU score: {1}".format(i,cwiou[i]))
```

```
overall IoU score for resnet50: 0.6818828536827107
class wise IoU score for resnet50
class 0 IoU score: 0.9381753223486402
class 1 IoU score: 0.38896461514365177
class 2 IoU score: 0.4714086558403688
class 3 IoU score: 0.7658375203379562
class 4 IoU score: 0.48149132116997995
class 5 IoU score: 0.767823852885584
class 6 IoU score: 0.9594786880527938
```

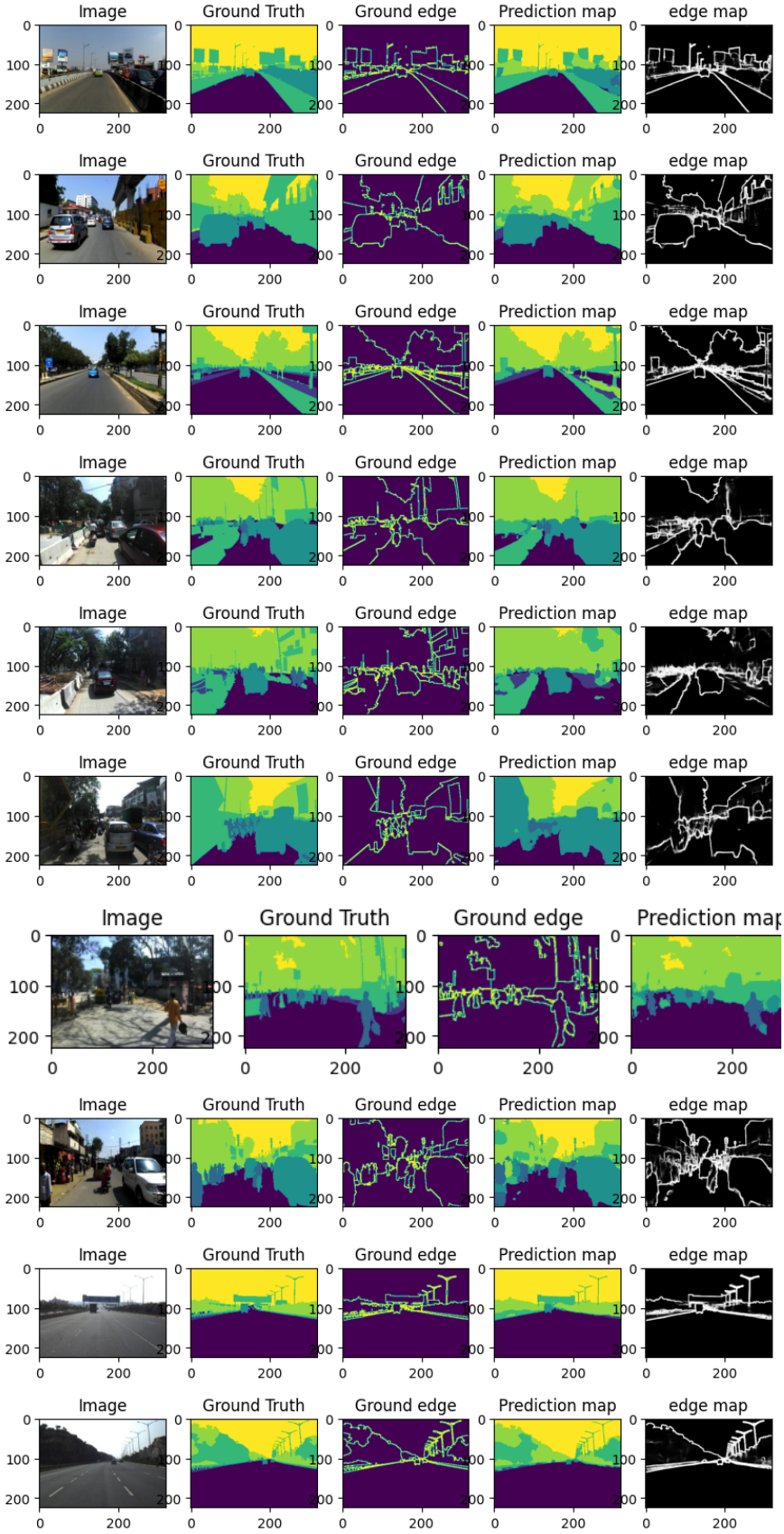
```
#Inference on some of the validation dataset
for i in range(10):
  X_val_samp=X_val_test[i]
  y_pred_samp,y_pred_edge_samp=model_resnet.predict(X_val_samp[np.newaxis,:,:,:])
  plt.figure(figsize=(10,15))
  plt.subplot(151)
  plt.imshow(X_val_samp/255.0)
  plt.title('Image')
  plt.subplot(152)
  plt.imshow(y_val_test[i])
  plt.title('Ground Truth')
  plt.subplot(153)
  plt.imshow(y_val_edge_test[i])
  plt.title("Ground edge")
  plt.subplot(154)
  plt.imshow(np.argmax(y_pred_samp[0],axis=-1))
  plt.title('Prediction map')
  plt.subplot(155)
  plt.imshow(y_pred_edge_samp[0],cmap='gray')
  plt.title('edge map')
```

```
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 33ms/step
1/1 [==============================] - 0s 33ms/step
1/1 [==============================] - 0s 32ms/step
1/1 [==============================] - 0s 32ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 34ms/step
1/1 [==============================] - 0s 35ms/step
1/1 [==============================] - 0s 37ms/step
```



```
for i in range(10,20):
    #original image
    #predicted segmentation map
  X_val_samp=X_val_test[i]
  y_pred_samp,y_pred_edge_samp=model_resnet.predict(X_val_samp[np.newaxis,:,:,:])
  plt.figure(figsize=(10,15))
  plt.subplot(151)
  plt.imshow(X_val_samp/255.0)
  plt.title('Image')
  plt.subplot(152)
  plt.imshow(y_val_test[i])
  plt.title('Ground Truth')
  plt.subplot(153)
  plt.imshow(y_val_edge_test[i])
  plt.title("Ground edge")
  plt.subplot(154)
  plt.imshow(np.argmax(y_pred_samp[0],axis=-1))
  plt.title('Prediction map')
  plt.subplot(155)
  plt.imshow(y_pred_edge_samp[0],cmap='gray')
  plt.title('edge map')
```

```
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 28ms/step
1/1 [==============================] - 0s 27ms/step
1/1 [==============================] - 0s 27ms/step
1/1 [==============================] - 0s 27ms/step
1/1 [==============================] - 0s 28ms/step
1/1 [==============================] - 0s 34ms/step
1/1 [==============================] - 0s 33ms/step
1/1 [==============================] - 0s 34ms/step
1/1 [==============================] - 0s 33ms/step
```



Training modified U-net-edge with efficientNet-b5

```
def encoder(inputs):
    skip_connections = []

    model = tf.keras.applications.EfficientNetB5(include_top=False, weights='imagenet', input_tensor=inputs)
    names = ["block2a_expand_activation", "block3a_expand_activation", "block4a_expand_activation", "block6a_expand_activation"
    for layer in model.layers:
        layer.trainable=False
    for name in names:
        skip_connections.append(model.get_layer(name).output)

    output = model.get_layer("top_activation").output
    return output, skip_connections

keras.backend.clear_session()
X_inp = Input(shape=(224,320,3))
# Encoder
c_5_5,conv_blocks = encoder(X_inp)
c_1_1,c_2_2,c_3_3,c_4_4=conv_blocks[0],conv_blocks[1],conv_blocks[2],conv_blocks[3]
# Decoder
dec_1_out=DecoderBlock(filters=256,name="decoder_1")((c_5_5,c_4_4))
dec_2_out=DecoderBlock(filters=128,name="decoder_2")((dec_1_out,c_3_3))
dec_3_out=DecoderBlock(filters=64,name="decoder_3")((dec_2_out,c_2_2))
dec_4_out=DecoderBlock(filters=32,name="decoder_4")((dec_3_out,c_1_1))
dec_5_upsamp=UpSampling2D(size=(2,2),interpolation='bilinear')(dec_4_out)
dec_5_conv1=Conv2D(filters=16,name="decoder_5_conv1",strides=1,kernel_size=3,padding="same")(dec_5_upsamp)
dec_5_bn1=BatchNormalization()(dec_5_conv1)
dec_5_act1=Activation("relu")(dec_5_bn1)
dec_5_conv2=Conv2D(filters=16,name="decoder_6_conv2",strides=1,kernel_size=3,padding="same")(dec_5_act1)
dec_5_bn2=BatchNormalization()(dec_5_conv2)
dec_5_act2=Activation("relu")(dec_5_bn2)

# Edge
conv_shape_out_1=Conv2D(filters=128,kernel_size=3,strides=1,padding="same")(dec_2_out)
upsample_out_1=UpsamplerBlock(filters=128,upsample_size=2,name="upsampler_1")(conv_shape_out_1)
res_out_1=ResBlock(filters=128,name="res_1")(upsample_out_1)
gcl_out_1=GclBlock(name="gcl_1")([res_out_1,dec_3_out])
upsample_out_2=UpsamplerBlock(filters=1,upsample_size=2,name="upsampler_2")(gcl_out_1)
res_out_2=ResBlock(filters=1,name="res_2")(upsample_out_2)
gcl_out_2=GclBlock(name="gcl_2")([res_out_2,dec_4_out])
upsample_out_3=UpsamplerBlock(filters=1,upsample_size=2,name="upsampler_3")(gcl_out_2)
res_out_3=ResBlock(filters=1,name="res_3")(upsample_out_3)
gcl_out_3=GclBlock(name='gcl_3')([res_out_3,dec_5_act2])
conv_shape_out_2=Conv2D(filters=1,kernel_size=3,strides=1,padding='same')(gcl_out_3)
sigmoid_out=Activation("sigmoid",name="sigmoid_out")(conv_shape_out_2)


#Final concatenation layers
concat_final_out=concatenate([sigmoid_out,dec_5_act2])
final_conv=Conv2D(filters=7,strides=1,kernel_size=3,padding="same")(concat_final_out)
softmax_out=Activation('softmax',name="softmax_out")(final_conv)

model_effnet=Model(inputs=X_inp,outputs=[softmax_out,sigmoid_out])
model_effnet.summary()

for layer in model_effnet.layers:
  layer.trainable=True
import segmentation_models as sm
optim=tf.keras.optimizers.Adam(0.001)

dice_loss=sm.losses.DiceLoss()
bin_ce=tf.keras.losses.BinaryCrossentropy()
categorical_ce=tf.keras.losses.SparseCategoricalCrossentropy()
biou=sm.metrics.IOUScore()
binary_loss=sm.losses.BinaryFocalLoss()
model_effnet.compile(optim,loss={"softmax_out":lambda y_true,y_pred: dice_loss(y_true,y_pred)+categorical_ce(y_true,y_pred),
                                 "sigmoid_out":lambda y_true,y_pred: dice_loss(y_true,y_pred)+bin_ce(y_true,y_pred)},
                     metrics={"softmax_out":miou,"sigmoid_out":biou})

callbacks = [
    tf.keras.callbacks.ModelCheckpoint('best_model_canet_t4.h5', save_weights_only=True, save_best_only=True, mode='min'),
    tf.keras.callbacks.TensorBoard('TBlog_canet_t4',histogram_freq=1),
        tf.keras.callbacks.ReduceLROnPlateau(monitor='val_loss',factor=0.5,patience=3,verbose=1,mode='min',min_lr=0.000001,min_
]
history_effnet_1 = model_effnet.fit(train_generator, epochs=40,steps_per_epoch=steps_per_epoch,validation_data=(X_val_test,(y_
```

```
t_loss: 1.0435 - val_sigmoid_out_loss: 0.6251 - val_softmax_out_miou: 0.6621 - val_sigmoid_out_iou_score: 0.4148 - lr: 0.0010
t_loss: 1.0636 - val_sigmoid_out_loss: 0.6448 - val_softmax_out_miou: 0.6371 - val_sigmoid_out_iou_score: 0.4075 - lr: 0.0010
t_loss: 1.0559 - val_sigmoid_out_loss: 0.8842 - val_softmax_out_miou: 0.6439 - val_sigmoid_out_iou_score: 0.3468 - lr: 0.0010

t_loss: 1.0733 - val_sigmoid_out_loss: 0.6245 - val_softmax_out_miou: 0.6528 - val_sigmoid_out_iou_score: 0.4121 - lr: 0.0010
t_loss: 1.0486 - val_sigmoid_out_loss: 0.6131 - val_softmax_out_miou: 0.6678 - val_sigmoid_out_iou_score: 0.4177 - lr: 5.0000e-04
t_loss: 1.0395 - val_sigmoid_out_loss: 0.6422 - val_softmax_out_miou: 0.6739 - val_sigmoid_out_iou_score: 0.4206 - lr: 5.0000e-04
t_loss: 1.0297 - val_sigmoid_out_loss: 0.6073 - val_softmax_out_miou: 0.6772 - val_sigmoid_out_iou_score: 0.4207 - lr: 5.0000e-04
t_loss: 1.0437 - val_sigmoid_out_loss: 0.6611 - val_softmax_out_miou: 0.6693 - val_sigmoid_out_iou_score: 0.4206 - lr: 5.0000e-04
t_loss: 1.0378 - val_sigmoid_out_loss: 0.6160 - val_softmax_out_miou: 0.6715 - val_sigmoid_out_iou_score: 0.4252 - lr: 5.0000e-04

t_loss: 1.0363 - val_sigmoid_out_loss: 0.6773 - val_softmax_out_miou: 0.6776 - val_sigmoid_out_iou_score: 0.4172 - lr: 5.0000e-04
t_loss: 1.0286 - val_sigmoid_out_loss: 0.6048 - val_softmax_out_miou: 0.6786 - val_sigmoid_out_iou_score: 0.4313 - lr: 2.5000e-04
t_loss: 1.0339 - val_sigmoid_out_loss: 0.6021 - val_softmax_out_miou: 0.6777 - val_sigmoid_out_iou_score: 0.4151 - lr: 2.5000e-04
t_loss: 1.0349 - val_sigmoid_out_loss: 0.5997 - val_softmax_out_miou: 0.6814 - val_sigmoid_out_iou_score: 0.4242 - lr: 2.5000e-04
t_loss: 1.0328 - val_sigmoid_out_loss: 0.5985 - val_softmax_out_miou: 0.6783 - val_sigmoid_out_iou_score: 0.4244 - lr: 2.5000e-04
t_loss: 1.0308 - val_sigmoid_out_loss: 0.5965 - val_softmax_out_miou: 0.6843 - val_sigmoid_out_iou_score: 0.4301 - lr: 2.5000e-04
t_loss: 1.0303 - val_sigmoid_out_loss: 0.5969 - val_softmax_out_miou: 0.6843 - val_sigmoid_out_iou_score: 0.4226 - lr: 2.5000e-04
t_loss: 1.0330 - val_sigmoid_out_loss: 0.5957 - val_softmax_out_miou: 0.6858 - val_sigmoid_out_iou_score: 0.4247 - lr: 2.5000e-04
t_loss: 1.0306 - val_sigmoid_out_loss: 0.5965 - val_softmax_out_miou: 0.6860 - val_sigmoid_out_iou_score: 0.4300 - lr: 2.5000e-04
t_loss: 1.0330 - val_sigmoid_out_loss: 0.6099 - val_softmax_out_miou: 0.6791 - val_sigmoid_out_iou_score: 0.4314 - lr: 2.5000e-04
t_loss: 1.0344 - val_sigmoid_out_loss: 0.6083 - val_softmax_out_miou: 0.6788 - val_sigmoid_out_iou_score: 0.4078 - lr: 2.5000e-04

t_loss: 1.0340 - val_sigmoid_out_loss: 0.5983 - val_softmax_out_miou: 0.6855 - val_sigmoid_out_iou_score: 0.4200 - lr: 2.5000e-04
t_loss: 1.0316 - val_sigmoid_out_loss: 0.5970 - val_softmax_out_miou: 0.6866 - val_sigmoid_out_iou_score: 0.4337 - lr: 1.2500e-04
t_loss: 1.0330 - val_sigmoid_out_loss: 0.5949 - val_softmax_out_miou: 0.6860 - val_sigmoid_out_iou_score: 0.4317 - lr: 1.2500e-04
```

```
pred = []
for i in range(len(X_val_test)):
  pred.append(model_effnet.predict(X_val_test[i][np.newaxis,:,:,:])[0])
pred1 = np.array(pred)
pred1 = pred1.reshape(pred1.shape[0],pred1.shape[2],pred1.shape[3],pred1.shape[4])
```

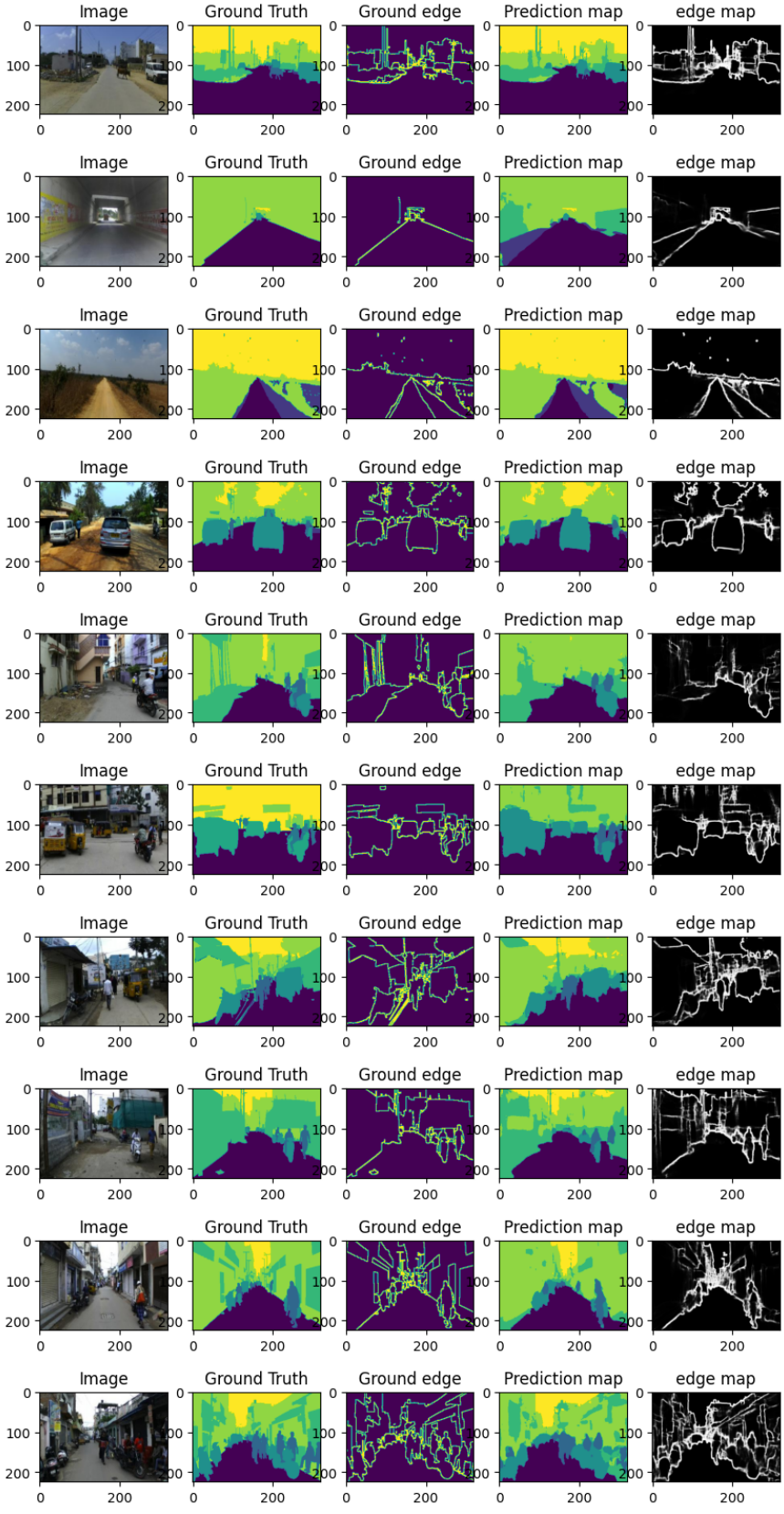
```
1/1 [==============================] - 0s 47ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 46ms/step
1/1 [==============================] - 0s 42ms/step
1/1 [==============================] - 0s 51ms/step
1/1 [==============================] - 0s 42ms/step
1/1 [==============================] - 0s 41ms/step
1/1 [==============================] - 0s 42ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 41ms/step
1/1 [==============================] - 0s 41ms/step
1/1 [==============================] - 0s 50ms/step
1/1 [==============================] - 0s 53ms/step
1/1 [==============================] - 0s 45ms/step
1/1 [==============================] - 0s 41ms/step
1/1 [==============================] - 0s 48ms/step
1/1 [==============================] - 0s 49ms/step
1/1 [==============================] - 0s 44ms/step
1/1 [==============================] - 0s 42ms/step
1/1 [==============================] - 0s 42ms/step
1/1 [==============================] - 0s 44ms/step
1/1 [==============================] - 0s 43ms/step
1/1 [==============================] - 0s 42ms/step
1/1 [==============================] - 0s 42ms/step
1/1 [==============================] - 0s 47ms/step
1/1 [==============================] - 0s 48ms/step
1/1 [==============================] - 0s 59ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 41ms/step
1/1 [==============================] - 0s 41ms/step
```

```
def class_wise_IoU(y_val, y_pred):
    class_iou = []
    n_classes = 7

    y_predi = np.argmax(y_pred, axis=-1)
    y_truei = y_val[:,:,:,0]

    for c in range(n_classes):
        TP = np.sum((y_truei == c) & (y_predi == c))
        FP = np.sum((y_truei != c) & (y_predi == c))
        FN = np.sum((y_truei == c) & (y_predi != c))
        IoU = TP / float(TP + FP + FN)
        if(float(TP + FP + FN) == 0):
          IoU=TP/0.001
        class_iou.append(IoU)
    return class_iou
cwiou = class_wise_IoU(y_val_test,pred1)
print("overall IoU score for efficientnetb5: {}".format(IoU(y_val_test,pred1)))
print("class wise IoU score for efficientnetb5")
for i in range(len(cwiou)):
    print("class {0} IoU score: {1}".format(i,cwiou[i]))
```

```
overall IoU score for efficientnetb5: 0.6945720384168055
class wise IoU score for efficientnetb5
class 0 IoU score: 0.9454070583840454
class 1 IoU score: 0.4186246108658751
class 2 IoU score: 0.48195468476291037
class 3 IoU score: 0.7801266516951506
class 4 IoU score: 0.5065766878338582
class 5 IoU score: 0.7716345542548593
class 6 IoU score: 0.9576800211209384
```

```
for i in range(10):
  X_val_samp=X_val_test[i]
  y_pred_samp,y_pred_edge_samp=model_effnet.predict(X_val_samp[np.newaxis,:,:,:])
  plt.figure(figsize=(10,15))
  plt.subplot(151)
  plt.imshow(X_val_samp/255.0)
  plt.title('Image')
  plt.subplot(152)
  plt.imshow(y_val_test[i])
  plt.title('Ground Truth')
  plt.subplot(153)
  plt.imshow(y_val_edge_test[i])
  plt.title("Ground edge")
  plt.subplot(154)
  plt.imshow(np.argmax(y_pred_samp[0],axis=-1))
  plt.title('Prediction map')
  plt.subplot(155)
  plt.imshow(y_pred_edge_samp[0],cmap='gray')
  plt.title('edge map')
```

```
1/1 [==============================] - 0s 48ms/step
1/1 [==============================] - 0s 50ms/step
1/1 [==============================] - 0s 50ms/step
1/1 [==============================] - 0s 43ms/step
1/1 [==============================] - 0s 48ms/step
1/1 [==============================] - 0s 49ms/step
1/1 [==============================] - 0s 43ms/step
1/1 [==============================] - 0s 42ms/step
1/1 [==============================] - 0s 45ms/step
1/1 [==============================] - 0s 44ms/step
```



```
for i in range(10,20):
  X_val_samp=X_val_test[i]
  y_pred_samp,y_pred_edge_samp=model_effnet.predict(X_val_samp[np.newaxis,:,:,:])
  plt.figure(figsize=(10,15))
  plt.subplot(151)
  plt.imshow(X_val_samp/255.0)
  plt.title('Image')
  plt.subplot(152)
  plt.imshow(y_val_test[i])
  plt.title('Ground Truth')
  plt.subplot(153)
  plt.imshow(y_val_edge_test[i])
  plt.title("Ground edge")
  plt.subplot(154)
  plt.imshow(np.argmax(y_pred_samp[0],axis=-1))
  plt.title('Prediction map')
  plt.subplot(155)
  plt.imshow(y_pred_edge_samp[0],cmap='gray')
  plt.title('edge map')
```

```
1/1 [==============================] - 0s 62ms/step
1/1 [==============================] - 0s 45ms/step
1/1 [==============================] - 0s 45ms/step
1/1 [==============================] - 0s 60ms/step
1/1 [==============================] - 0s 54ms/step
1/1 [==============================] - 0s 53ms/step
1/1 [==============================] - 0s 44ms/step
1/1 [==============================] - 0s 46ms/step
1/1 [==============================] - 0s 53ms/step
1/1 [==============================] - 0s 52ms/step
```